Package 'mc'

February 1, 2016

Version 1.0.2

Author Rex Cheung <rccheung@ucdavis.edu>, Teresa Filshtein <teresa.filshtein@gmail.com>, Norm Matloff <matloff@cs.ucdavis.edu>, and Ozan Sonmez <osonmez@ucdavis.edu>

Maintainer Teresa Filshtein <teresa.filshtein@gmail.com>

Date 2016-02-01

Title Markov Chains

Description A new package that performs various calculations on Markov Chains

Depends

Imports markovchain, lattice, methods

Suggests

LazyLoad no

License GPL (>= 2)

Repository CRAN

NeedsCompilation no

Date/Publication 2016-02-01 12:00:00

R topics documented:

absorb *Absorbing Markov Chain*

Description

Finds key features of a Discrete Finite absorbing Markov Chain. Future updates will include the infinite case

Usage

```
absorb.mc(pijdef, type, tol = 1e-06, ...)
```

Arguments

`pijdef`	The transition probabilities, either in matrix form or a function For now only matrix form, until the infinite case is incorporated
`type`	Type of markov chain, either 'discrete' or 'continuous'.
`tol`	A positive scalar for error tolerance for infinite markov chain approximation.
`...`	Additional argument for continuous markov chain type.

Details

This function generates key features and common output for an absorbing Markov Chain. An error is returned if the input matrix is not absorbing. A Markov Chain is an absorbing chain if 1) At least one state is absorbing 2) All non-absorbing states are transient.

Value

Object of class "mc", with components

`pijdef`	The original transition matrix
`absorb`	TRUE,Confirms it is an absorbing chain
`astates`	absorbing states
`tstates`	transient states
`canonicalForm`	canonical form of the transition matrix
`steady.state`	stationary distribution
`FundamentalMatrix`	fundamental matrix
`Qmat`	Q matrix, the sub matrix of transient to transient states
`Rmat`	R matrix, the sub matrix of transient to absorbing states
`ExpectedHits`	Expected number of hits in state i before getting absorbed
`AbsorbProb`	A matrix of absorption probalities. Columns represent the absorbing states, rows represent transient states. The i,j entry represents the probablilty of being absorbed into absorbing state j given you are in state i

Author(s)

Teresa Filshtein <teresa.filshtein@gmail.com>,Ozan Sonmez <osonmez@ucdavis.edu>, Rex Cheung <rccheung@ucdavis.edu>, and Norm Matloff <matloff@cs.ucdavis.edu>

Examples

```
#Discrete Chain
#Finite state


drunkard = t(matrix(c(1,rep(0,4),.5,0,.5,0,0,0,.5,0,.5,0,0,0,.5,0,.5,
                    rep(0,4),1),nrow = 5))
absorb.mc(drunkard)$ExpectedHits
absorb.mc(drunkard)$AbsorbProb
```

`expectstep` *Expected Steps/Time*

Description

Finds the expected steps/time from one state to another.

Usage

```
expectstep(pijdef, type, rows, columns, tol = 1e-6, ...)
steps.fin(pijdef, type, rows, columns)
steps.inf(pijdef, type, rows, columns, tol = 1e-06)
timefin(rows, columns, ...)
timeinf(rows, columns, tol = 1e-6, ...)
```

Arguments

`pijdef`	The transition probabilities, either in matrix form (for finite states Markov chain) or a function (for infinite states Markov chain).
`type`	Type of Markov chain, either 'discrete' or 'continuous'.
`rows`	A numeric vector of initial states of interest.
`columns`	A numeric vector of destination states of interest.
`tol`	A positive scalar for error tolerance for infinite state Markov chain approximation.
`...`	Additional argument for continuous type Markov chain (see details).

Details

This function finds the expected steps it takes to reach from one state to another in an irreducible Markov chain. User should input a matrix defining the transition probabilities for a finite state Markov chain, or a function specifying the probabilities for infinite state Markov chain (see example).

The expected steps for the infinite state Markov chain is calculated via approximation. Each time we assume a finite number of states and calculate the expected number of steps, then increase the number of states and recalculate the expected steps, until the one norm of the difference in the step matrices is less than the tolerate level.

For infinite state Markov chain, the forward probability should be smaller than the backward probability (except at the initial states), i.e. p_ij < p_ik for j > k. This is to guarantee the chain will converge. If such condition is violated, only the expected steps from states i to states j for all i < j can be calculated, or an error will be returned otherwise.

For continuous type Markov chain, user has the option of either supplying the holding rate at each state using the argument *qidef*, or simply supplying the transition rate matrix using the argument *transrate*. If supplying holding rates, the probability transition matrix must also be given (see example below). Note *diag(transrate) = -qidef*.

Value

Object of class "mc", with components

`pijdef`	The input transition probability definition if supplied.
`steps`	(For discrete chains) The expected steps in a matrix form, with rows specifying the initial states and columns specifying the destination states.
`time`	(For continuous chains) The expected time in a matrix form, with rows specifying the initial states and columns specifying the destination states.
`holding.rates`	If the chain is of continuous type and user supplied holding rates, it will also return the holding rates
`transrate`	If the chain is of continuous type and user supplied transition rates, it will also return the transition rates

Author(s)

Rex Cheung <rccheung@ucdavis.edu>, Teresa Filshtein <teresa.filshtein@gmail.com>, Norm Matloff <matloff@cs.ucdavis.edu>, and Ozan Sonmez <osonmez@ucdavis.edu>

Examples

```
##Discrete Markov Chain
#Finite state
P = matrix(rep(0.5, 9), ncol = 3)
P[1,3] = 0; P[2,2] = 0; P[3,1] = 0
expectstep(P, type = 'discrete', rows = 1:3, columns = 1:3)

#Infinite states
pijdef <- function(i,j){
  if (i == 1 && j == 2) return(0.4)
  if (i == 1 && j == 1) return(0.6)
  if ((i-j) == -1) return(0.4)
  if ((i-j) == 1) return(0.6)
  0
}
expectstep(pijdef, type = 'discrete', rows = c(2,4,5), columns = c(2,3,7,9))

## Not run:
##Continuous Markov Chain
#Finite States
#With probability transition matrix and holding rates
qidef = c(0.25, 0.175, 0.08)
pijdef = matrix(c(0, (1/20)/((1/20)+(1/8)), 0, 1, 0, 1, 0, (1/8)/((1/20)+(1/8)), 0), nrow = 3)
expectstep(pijdef, 'continuous', rows = 1:3, columns = 1:3, qidef = qidef)
#With transition rates
Q = matrix(c(-0.25,0.05,0,0.25,-0.175,0.08,0,0.125,-0.08), nrow = 3)
expectstep(type = 'continuous', rows = 1:3, columns = 1:3, transrate = Q)
```

```
#Infinite States (Birth death process)
#With Transition rates
transrate = function(i,j){
  if(i == 1 && j == 1) return(-1)
  if(i == 1 && j == 2) return(1)
  if(i == 2 && j == 1) return(1)
  if(i == 2 && j == 2) return(-4/3)
  if(i == 2 && j == 3) return(1/3)
  if(j - i == 1) return(1/(i+1)) #Foward
  if(j - i == -1) return(1/(i-1)) #Backward
  if(j - i == 0) return(-1/(i-1)-1/(i+1)) #Itself
  0
}
expectstep(pijdef = NULL, type = 'continuous', rows = 1:3, columns = 1:3, transrate = transrate, tol = 0.5)

#With probability transition matrix and holding rates
holding = function(i){
  if(i == 1) return(1)
  return(1/(i-1) + 1/(i+1))
}
pijdef = function(i,j){
  if(i == 1 && j == 2) return(1)
  if(i == 1 && j == 1) return(0)
  if(j - i == 1) return((1/(i+1))/((1/(i-1)) + (1/(i+1))))
  if(j - i == 0) return(0)
  if(j - i == -1) return((1/(i-1))/((1/(i-1)) + (1/(i+1))))
  0
}
expectstep(pijdef = pijdef, type = 'continuous', rows = 1:3, columns = 1:3, qidef = holding, tol = 0.5)

## End(Not run)
```

`mc.simulation` *Markov Chain Simulation*

Description

Simulates discrete markov chain with state space

Usage

```
mc.simulation(pijdef, type, N, initial.state,...)
```

Arguments

`pijdef`	The transition probabilities, either in matrix form or a function For now only matrix form, until the infinite case is incorporated
`type`	Type of markov chain, either 'discrete' or 'continuous'.
`N`	Number of steps of the markov chain being simulated
`initial.state`	Initial state that the chain will start from
`...`	Additional argument for continuous markov chain type.

Details

This function will simulate discrete markov chain for a given transition probability matrix, if the initial state is not specified then it will be randomly generated from the state space.

Value

`pijdef`	The original transition matrix
`simulated.states`	N Simulated states based on the transition matrix

Author(s)

Teresa Filshtein <teresa.filshtein@gmail.com>, Ozan Sonmez <osonmez@ucdavis.edu>, Rex Cheung <rccheung@ucdavis.edu>, and Norm Matloff <matloff@cs.ucdavis.edu>

Examples

```
t.matrix = t(matrix(c(0.1,0.2,0.7,0.4,0.5,0.1,0,0.4,0.6),3,3))
mc.simulation(t.matrix, "discrete", 100, 1)$simulated.states
```

`plotstep` *Plot heatmap of expected steps*

Description

Plots the expected steps for finite state irreducible Markov chain in a heatmap

Usage

```
plotstep(pijdef, type, init = 1:nrow(pijdef), dest = 1:ncol(pijdef), ...)
```

Arguments

`pijdef`	The transition probabilities in matrix form.
`type`	Type of Markov chain, either 'discrete' or 'continuous'.
`init`	Set of desired initial states (default to all states).
`dest`	Set of desired destination states (default to all states).
`...`	Additional arguments for continuous Markov chain (see details).

Details

This function plots the heatmap of the expected steps of a finite state irreducible Markov chain. The user can supply the desired initial states and/or destination states using the arguments *init* and/or *dest* respectively. The function will plot the heatmap for the expected steps between states if these arguments are not supplied.

For continuous type markov chain, users must supply either both probability transition matrix (*pijdef*) and holding rates (*qidef*), or the transition rate matrix (*transrate*) itself. See ?expectstep for more details.

Value

A plot of the expected steps for the desired initial and destination states in the form of a heatmap.

Author(s)

Rex Cheung <rccheung@ucdavis.edu>, Teresa Filshtein <teresa.filshtein@gmail.com>, Norm Matloff <matloff@cs.ucdavis.edu>, and Ozan Sonmez <osonmez@ucdavis.edu>

Examples

```
#Discrete Markov chain
P = matrix(rep(0.5, 9), ncol = 3)
P[1,3] = 0; P[2,2] = 0; P[3,1] = 0
plotstep(P, 'discrete')
plotstep(P, 'discrete', init = c(1,3))
plotstep(P, 'discrete', init = c(1,3), dest = c(2,3))

#Continuous Markov chain
#With probability transition matrix and holding rates
qidef = c(0.25, 0.175, 0.08)
pijdef = matrix(c(0, (1/20)/((1/20)+(1/8)), 0, 1, 0, 1, 0, (1/8)/((1/20)+(1/8)), 0), nrow = 3)
plotstep(pijdef, type = 'continuous', init = 1:2, dest = 1:3, qidef = qidef)
#With transition rates
Q = matrix(c(-0.25,0.05,0,0.25,-0.175,0.08,0,0.125,-0.08), nrow = 3)
plotstep(type = 'continuous', init = 1:2, dest = 1:3, transrate = Q)
```

`stationary` *Stationary Distribution*

Description

Finds stationary distribution of an irreducible Markov chain.

Usage

```
stationary(pijdef, type, tol = 1e-06, ...)
findpil.fin(pijdef)
findpil.inf(pijdef, tol = 1e-06)
findpicont.fin(...)
findpicont.inf(tol = 1e-6,...)
```

Arguments

`pijdef`	The transition probabilities, either in matrix form (for finite states Markov chain) or a function (for infinite states Markov chain).
`type`	Type of Markov chain, either ’discrete’ or ’continuous’.
`tol`	A positive scalar for error tolerance for infinite state Markov chain approximation.
`...`	Additional argument for continuous type Markov chain (see details).

Details

This function finds the stationary distribution of a given transition probability definition. User should input a matrix defining the transition probabilities for a finite state Markov chain, or a function specifying the probabilities for infinite state Markov chain (see example).

The stationary distribution for the infinite states Markov chain is calculated via approximation. Each time we assume a finite number of states and calculate the stationary distributions, then increase the number of states and recalculate the stationary distributions, until the difference in between the distributions is less than the tolerate level.

For infinite state Markov chain, the forward probability should be smaller than the backward probability (except at the initial state), i.e. p_ij < p_ik for j > k, i > 1. This is to guarantee the chain will converge. An error will return if this condition is violated.

If the chain is of continuous type, user has the option of either supplying the holding rate at each state using the argument *qidef*, or simply supplying the transition rate matrix using the argument *transrate*. If supplying holding rates, the probability transition matrix must also be given (see example below).

Value

Object of class "mc", with components

`pijdef`	The input transition probability definition
`stationary.distribution`	A numeric vector of probabilities for the stationary distribution
`holding.rates`	If the chain is of continuous type and user supplied holding rates, it will also return the holding rates
`transrate`	If the chain is of continuous type and user supplied transition rates, it will also return the transition rates

Author(s)

Rex Cheung <rccheung@ucdavis.edu>, Teresa Filshtein <teresa.filshtein@gmail.com>, Norm Matloff <matloff@cs.ucdavis.edu>, and Ozan Sonmez <osonmez@ucdavis.edu>

Examples

```
##Discrete Markov Chains
#Finite state
P = matrix(rep(0.5, 9), ncol = 3)
P[1,3] = 0; P[2,2] = 0; P[3,1] = 0
stationary(P, type = 'discrete')

#Infinite states
pijdef <- function(i,j){
 if (i == 1 && j == 2) return(0.3)
 if (i == 1 && j == 1) return(0.7)
 if ((i-j) == -1) return(0.3)
 if ((i-j) == 1) return(0.7)
 0
}
stationary(pijdef, type = 'discrete')

## Not run:
##Continuous Markov Chains
```

```
#Finite States
#With probability transition matrix and holding rates
qidef = c(0.25, 0.175, 0.08)
pijdef = matrix(c(0, (1/20)/((1/20)+(1/8)), 0, 1, 0, 1, 0, (1/8)/((1/20)+(1/8)), 0), nrow = 3)
stationary(pijdef = pijdef, type = 'continuous', qidef = qidef)
#With transition rates
Q = matrix(c(-0.25, 0.25, 0, 0.05, -0.175, -.125, 0, 0.08, -0.08), nrow = 3)
stationary(type = 'continuous', transrate = Q)

#Infinite States (Birth death process)
#With transition rates
transrate = function(i,j){
  if(i == 1 && j == 1) return(-1)
  if(i == 1 && j == 2) return(1)
  if(i == 2 && j == 1) return(1)
  if(i == 2 && j == 2) return(-4/3)
  if(i == 2 && j == 3) return(1/3)
  if(j - i == 1) return(1/(i+1)) #Foward
  if(j - i == -1) return(1/(i-1)) #Backward
  if(j - i == 0) return(-1/(i-1)-1/(i+1)) #Itself
  0
}
#The diagonal elements are the negative of holding rates at each state
stationary(type = 'continuous', transrate = transrate, tol = 1e-1)

#With probability transition matrix and holding rates
qidef = function(i){
  if(i == 1) return(1)
  return(1/(i-1) + 1/(i+1))
}
pijdef = function(i,j){
  if(i == 1 && j == 2) return(1)
  if(i == 1 && j == 1) return(0)
  if(j - i == 1) return((1/(i+1))/((1/(i-1)) + (1/(i+1))))
  if(j - i == 0) return(0)
  if(j - i == -1) return((1/(i-1))/((1/(i-1)) + (1/(i+1))))
  0
}
stationary(type = 'continuous', pijdef = pijdef, qidef = qidef, tol = 1e-1)

## End(Not run)
```

Index